Mathematics of Data: From Theory to Computation

Prof. Volkan Cevher
volkan.cevher@epfl.ch

Lecture 9: Deep Learning III

Laboratory for Information and Inference Systems (LIONS)
École Polytechnique Fédérale de Lausanne (EPFL)

EE-556 (Fall 2020)

lions@epfl HASLERSTIFTUNG Google AI SDSC ZEISS FNSNF Fonds National Suisse Schweizerischer Nationalfonds Fondo Nazionale Svizzero Swiss National Science Foundation erc EPFL

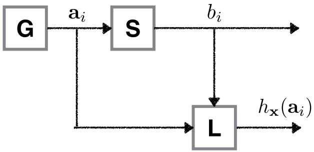
License Information for Mathematics of Data Slides

- This work is released under a Creative Commons License with the following terms:
- **Attribution**
  - The licensor permits others to copy, distribute, display, and perform the work. In return, licensees must give the original authors credit.
- **Non-Commercial**
  - The licensor permits others to copy, distribute, display, and perform the work. In return, licensees may not use the work for commercial purposes – unless they get the licensor's permission.
- **Share Alike**
  - The licensor permits others to distribute derivative works only under a license identical to the one that governs the licensor's work.
- Full Text of the License

Outline

- Scalable non-convex optimization with emphasis on deep learning

Recall: The general setting...



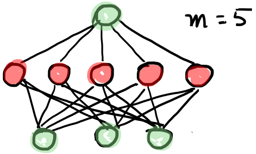
Definition (Optimization formulation)

The deep-learning training problem is given by

$$\mathbf{x}^\star_{\mathsf{DL}} \in \arg\min_{\mathbf{x}\in\mathcal{X}} \left\{ f(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} L(h_{\mathbf{x}}(\mathbf{a}_i), b_i) \right\},$$

where $\mathcal{X}$ denotes the constraints on the parameters.

◦ A single hidden layer neural network with params $\mathbf{x} := [\mathbf{X}_1, \mathbf{X}_2, \mu_1, \mu_2]$



$$h_{\mathbf{x}}(\mathbf{a}) := \underbrace{\left[\ \mathbf{X}_2\ \right] \overset{\text{activation}}{\sigma} \left(\overset{\text{weight}}{\left[\ \mathbf{X}_1\ \right]} \overset{\text{input}}{\left[\mathbf{a}\right]} + \overset{\text{bias}}{\left[\mu_1\right]} \right)}_{\text{hidden layer = learned features}} + \overset{\text{bias}}{\left[\mu_2\right]}$$

Towards training with neural networks

◦ What do we have at hand?

1. The optimization objective $f(\mathbf{x})$ from multi-layer, multi-class, convolutions, transformers, etc.
2. First-order gradient via backpropagation $\nabla f(\mathbf{x})$

Towards training with neural networks

◦ What do we have at hand?

1. The optimization objective $f(\mathbf{x})$ from multi-layer, multi-class, convolutions, transformers, etc.
2. First-order gradient via backpropagation $\nabla f(\mathbf{x})$

◦ Barriers to training of neural networks:

1. Curse-of-dimensionality
2. Non-convexity
3. Ill-conditioning

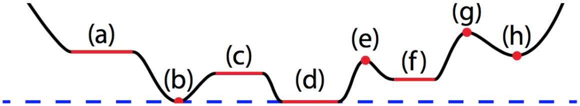


Figure: A non-convex function. (a) and (c) are plateaus, (b) and (d) are global minima, (f) and (h) are local minima, (e) and (g) are local maxima. [17]

Towards training with neural networks

◦ What do we have at hand?

1. The optimization objective $f(\mathbf{x})$ from multi-layer, multi-class, convolutions, transformers, etc.
2. First-order gradient via backpropagation $\nabla f(\mathbf{x})$

◦ Barriers to training of neural networks:

1. Curse-of-dimensionality $\rightarrow$ first-order methods, see lecture 3
2. Non-convexity $\rightarrow$ stochasticity + momentum, this lecture
3. Ill-conditioning $\rightarrow$ adaptive gradient methods, this lecture

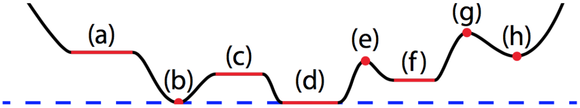


Figure: A non-convex function. (a) and (c) are plateaus, (b) and (d) are global minima, (f) and (h) are local minima, (e) and (g) are local maxima. [17]

Stochastic Gradient Descent (SGD) and some key variants

Vanilla (Minibatch) SGD
Input: Stochastic gradient oracle $\mathbf{g}$, initial point $\mathbf{x}^0$, step size γ_k
1. For $k = 0, 1, \ldots$:   obtain the (minibatch) stochastic gradient $\mathbf{g}^k$ update $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - \gamma_k \mathbf{g}^k$

Stochastic Gradient Descent (SGD) and some key variants

Vanilla (Minibatch) SGD
Input: Stochastic gradient oracle $\mathbf{g}$, initial point $\mathbf{x}^0$, step size γ_k
1. For $k = 0, 1, \ldots$:   obtain the (minibatch) stochastic gradient $\mathbf{g}^k$ update $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - \gamma_k \mathbf{g}^k$

Perturbed Stochastic Gradient Descent [13]
Input: Stochastic gradient oracle $\mathbf{g}$, initial point $\mathbf{x}^0$, step size γ_k
1. For $k = 0, 1, \ldots$:   sample noise ξ uniformly from unit sphere   update $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - \gamma_k (\mathbf{g}^k + \xi)$

Stochastic Gradient Descent (SGD) and some key variants

Vanilla (Minibatch) SGD
Input: Stochastic gradient oracle $\mathbf{g}$, initial point $\mathbf{x}^0$, step size γ_k
1. For $k = 0, 1, \ldots$: obtain the (minibatch) stochastic gradient $\mathbf{g}^k$ update $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - \gamma_k \mathbf{g}^k$

Perturbed Stochastic Gradient Descent [13]
Input: Stochastic gradient oracle $\mathbf{g}$, initial point $\mathbf{x}^0$, step size γ_k
1. For $k = 0, 1, \ldots$: sample noise ξ uniformly from unit sphere update $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - \gamma_k (\mathbf{g}^k + \xi)$

***Stochastic Gradient Langevin Dynamics [38]**
Input: Stochastic gradient oracle g, initial point $\mathbf{x}^0$, step size γ_k
1. For $k = 0, 1, \ldots$ sample noise ξ standard Gaussian update $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^l - \gamma_k \mathbf{g}^k + \sqrt{2\gamma_k}\xi$

Basic questions:

1. Does SGD converge with probability 1?
2. Does SGD avoid non-minimum points with probability 1?
3. How fast does SGD converge to local minimizers?
4. Can SGD converge to global minimizers?

Critical points

Recall (Classification of critical points)

Let $f : \mathbb{R}^d \to \mathbb{R}$ be twice differentiable and let $\bar{\mathbf{x}}$ be a critical point. Let $\{\lambda_i\}_{i=1}^d$ be the eigenvalues of the hessian $\nabla^2 f(\bar{\mathbf{x}})$, then

- $\lambda_i > 0$ for all $i \Rightarrow \bar{\mathbf{x}}$ is a local minimum
- $\lambda_i < 0$ for all $i \Rightarrow \bar{\mathbf{x}}$ is a local maximum
- $\lambda_i > 0$, $\lambda_j < 0$ for some i, j and $\lambda_i \neq 0$ for all $i \Rightarrow \bar{\mathbf{x}}$ is a saddle point
- Other cases $\Rightarrow$ inconclusive

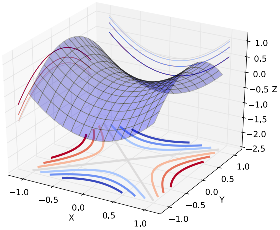


Figure: Minmax saddle ($\lambda_i \neq 0$ for all i)

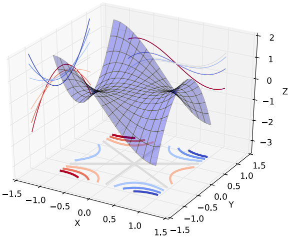


Figure: Monkey saddle ($\lambda_i = 0$ for some i)

The strict saddle property

Definition (Strict saddle)

A twice differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is $(\alpha, \beta, \epsilon, \delta)$-strict saddle if for any point $\mathbf{x}$ at least one of the following is true

1. $\|\nabla f(\mathbf{x})\| \geq \epsilon$.
2. $\lambda_{\min}\left(\nabla^2 f(\mathbf{x})\right) \leq -\beta$.
3. *There is a local minimum $\mathbf{x}^*$ such that $\|\mathbf{x} - \mathbf{x}^*\| \leq \delta$ and the function f restricted to a 2δ neighborhood of $\mathbf{x}^*$ is α strongly convex.*

(Informal)

For any point whose gradient is small, it is either close to a local minimum, or is a saddle point (or local maximum) with a significant negative eigenvalue.

Q1: Does SGD converge?

∘ SGD converges to the critical points of f as $N \to \infty$.

1. GD converges from any intialization with constant step-size and full gradients
2. With probability 1, (P)SGD does not converge with constant step-size γ [4, 33]
3. With probability 1, SGD converges with vanishing step-size if $\mathbf{x}^k$ is bounded with probability 1 [29, 4]

Boundedness is not required (Theorem 1 of [30])

Assume Lipschitzness, **sublevel regularity**, $\mathbb{E}\|\mathbf{g}\|^q \leq \sigma^q$ and $\sum_k \gamma_k^{1+q/2} < \infty$ $(q \geq 2)$. Then, $\mathbf{x}^k$ converges with probability 1.

Q2: Does SGD avoid saddle points?

◦ SGD avoids strict saddles $(\lambda_{\min}(\nabla^2 f(\mathbf{x}^*)) < 0)$

1. GD avoids strict saddles from almost all initializations [24]

2. With probability $1 - \zeta$, PSGD with constant γ escapes strict saddles after $\Omega\left(\log(1/\zeta)/\gamma^2\right)$ iterations [14]

    - However, SGD does not converge with constant γ
    - We cannot take $\zeta = 0$

SGD avoids traps almost surely (Theorem 3 of [30])

Assume bounded uniformly exciting noise and $\gamma_k = \mathcal{O}\left(\frac{1}{k^\kappa}\right)$ for $\kappa \in (0, 1]$. Then, SGD avoids strict saddles from any initial condition with probability 1.

Q3: How fast does SGD converge to local minimizers?

◦ SGD remains close to Hurwicz minimizers (i.e., $\mathbf{x}^\star : \lambda_{\min}(\nabla^2 f(\mathbf{x}^\star)) > 0$)

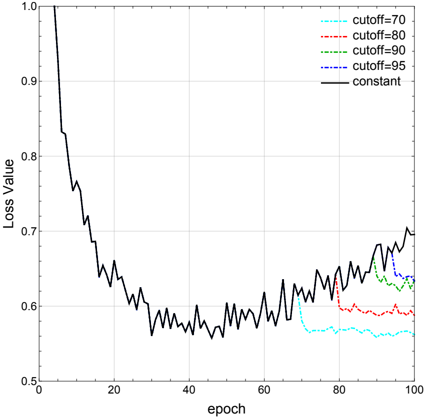
1. SGD with constant γ can obtain objective value ϵ-close to a Hurwicz minimizer in $\mathcal{O}(1/\epsilon^2)$-iterations [14, 15]
   - However, SGD does not converge with constant γ
   - Need averaging which is problematic in non-convex optimization

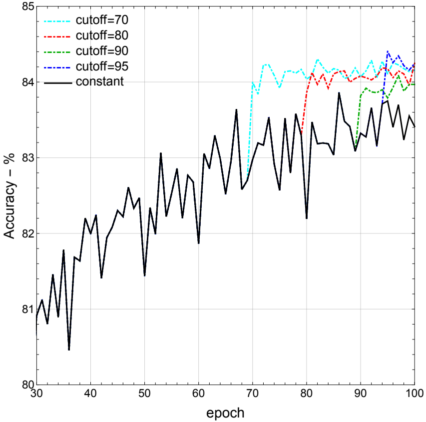
Using a vanishing step-size helps! (Theorem 4 of [30])

Using $\gamma_k = \mathcal{O}\left(\frac{1}{k}\right)$, SGD enjoys a $\mathcal{O}\left(\frac{1}{k}\right)$ convergence rate in objective value.

Using $1/k$ step-size decrease helps in practice

◦ ResNet training at different cool-down cut-offs





Q4: Can SGD converge to global minimizers?

- A few phenomena about neural networks [42]:
  - Deep neural networks can fit random labels
  - First-order methods can find global minimizers

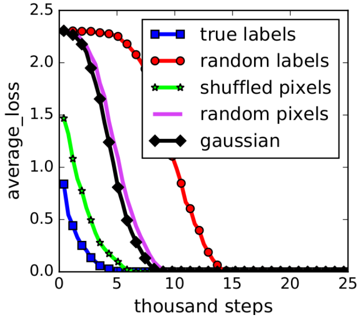


Figure: DNN Training curves on CIFAR10, from [42]

Q4: Can SGD converge to global minimizers?

- A few phenomena about neural networks [42]:
  - Deep neural networks can fit random labels
  - First-order methods can find global minimizers

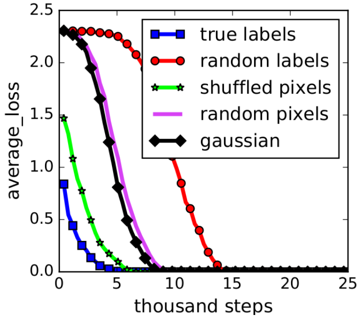


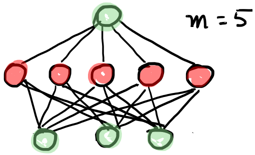
Figure: DNN Training curves on CIFAR10, from [42]

- Overparametrization can explain these mysteries!

Overparametrization

Number of parameters $\gg$ number of training data.

GD finds global minimizers of overparametrized networks



$$h_{\mathbf{x}}(\mathbf{a}) := \left[\ \mathbf{X}_2\ \right] \underbrace{\overset{\text{activation}}{\sigma} \left(\left[\ \overset{\text{weight}}{\mathbf{X}_1}\ \right] \left[\overset{\text{input}}{\mathbf{a}}\right] + \left[\overset{\text{bias}}{\mu_1}\right] \right)}_{\text{hidden layer } = \text{ learned features}} + \left[\overset{\text{bias}}{\mu_2}\right]$$

Theorem (Linear convergence of Gradient Descent [10])

- $f(\mathbf{a}; \mathbf{X}_1, \mathbf{X}_2)$: *1-hidden-layer network with width* m,*hidden layer weights* $\mathbf{X}_1$, *output layer weights* $\mathbf{X}_2$ *and ReLu activation.*
- $m = \Omega(\frac{n^6}{\delta^3})$ *where* n =*number of samples.*
- $\mathbf{X}_1^0$ *is initialized with a normal distribution,* $\mathbf{X}_2^0 \sim$ *Unif*$[-1, 1]^m$.
- *Stepsize* $\eta = O(n^{-2})$.

With probability at least $1 - \delta$, *for the empirical risk* R_n *we have*

$$R_n(\beta_t, W_t, b_t) \leq (1 - \eta)^t R_n(\beta_0, W_0, b_0) \tag{1}$$

Optimization landscape of overparametrized neural networks

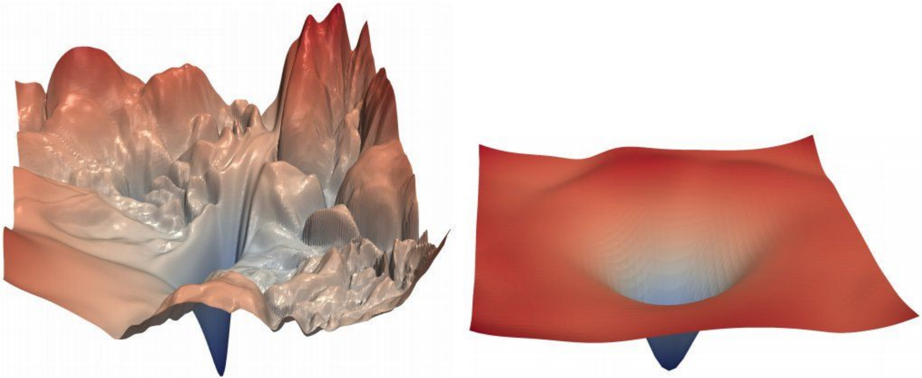


Figure: Intuitive comparison, loss landscape with few parameters (left) vs overparametrized regime (right). From [27], originally skip connections vs. no skip connections

Overparametrization is an active area of research

Reference	Number of parameters	Depth d	Result
[19, 20, 16]	$\tilde{\Omega}(n)$	1, 2	Existence of zero error
[40, 18, 31]	$\tilde{\Omega}(n)$	Any d	Existence of zero error
[28]	$\tilde{\Omega}(\text{poly}(n))$	1	(S)GD global convergence
[10]	$\tilde{\Omega}(n^6)$	1	(S)GD global convergence
[34]	$\tilde{\Omega}(n^2)$	1	(S)GD global convergence
[2, 44]	$\tilde{\Omega}(\text{poly}(n, d))$	Any d	(S)GD global convergence
[9]	$\tilde{\Omega}(n^8 2^{O(d)})$	Any d	(S)GD global convergence
[45]	$\tilde{\Omega}(n^8 d^1 2)$	Any d	(S)GD global convergence
[21]	$\tilde{\Omega}(n)$	Any d	(S)GD global convergence

Table: Summary of results on overparametrization. Minimum number of parameters required as a function of data size n and depth d. The result is classified either as *Existence* i.e., there exists a neural network achieving zero error on the data, or *(S)GD global convergence* i.e., (S)GD converges to zero training error, a much stronger condition.

Stochastic adaptive first-order methods

Adaptive methods

Stochastic adaptive methods converge without knowing the smoothness constant.

They do so by making use of the information from stochastic gradients and their norms.

Variable metric stochastic gradient descent algorithm

Variable metric stochastic gradient descent algorithm

1. Choose $\mathbf{x}^0 \in \mathbb{R}^p$ as a starting point and $\mathbf{H}_0 \succ 0$.
2. For $k = 0, 1, \cdots$, perform:

$$\begin{cases} \mathbf{d}^k & := -\mathbf{H}_k^{-1}\mathbf{g}^k, \\ \mathbf{x}^{k+1} & := \mathbf{x}^k + \alpha_k \mathbf{d}^k, \end{cases}$$

where $\alpha_k \in (0, 1]$ is a given step size.

3. Update $\mathbf{H}_{k+1} \succ 0$ if necessary.

Variable metric stochastic gradient descent algorithm

Variable metric stochastic gradient descent algorithm

1. Choose $\mathbf{x}^0 \in \mathbb{R}^p$ as a starting point and $\mathbf{H}_0 \succ 0$.
2. For $k = 0, 1, \cdots,$ perform:

$$\begin{cases} \mathbf{d}^k & := -\mathbf{H}_k^{-1}\mathbf{g}^k, \\ \mathbf{x}^{k+1} & := \mathbf{x}^k + \alpha_k \mathbf{d}^k, \end{cases}$$

where $\alpha_k \in (0, 1]$ is a given step size.

3. Update $\mathbf{H}_{k+1} \succ 0$ if necessary.

Common choices of the variable metric $\mathbf{H}_k$

- $\mathbf{H}_k := \lambda_k \mathbf{I}$ $\Longrightarrow$ stochastic gradient descent method.
- $\mathbf{H}_k := \mathbf{D}_k$ (a positive diagonal matrix) $\Longrightarrow$ stochastic adaptive gradient methods.

Adaptive gradient methods

Intuition

Adaptive gradient methods adapt locally by setting $\mathbf{H}_k$ as a function of past stochastic gradient information.

Adaptive gradient methods

Intuition

Adaptive gradient methods adapt locally by setting $\mathbf{H}_k$ as a function of past stochastic gradient information.

- Roughly speaking, $\mathbf{H}_k = \text{function}(\mathbf{g}^1, \mathbf{g}^2, \cdots, \mathbf{g}^k)$
- Some well-known examples:

AdaGrad [11]

$$\mathbf{H}_k = \sqrt{\sum_{t=1}^{k} \mathbf{g}^k {\mathbf{g}^k}^\top}$$

RmsProp [35]

$$\mathbf{H}_k = \sqrt{\beta \mathbf{H}_{k-1} + (1-\beta)\text{diag}(\mathbf{g}^k)^2}$$

ADAM [23]

$$\hat{\mathbf{H}}_k = \beta \hat{\mathbf{H}}_{k-1} + (1-\beta)\text{diag}(\mathbf{g}^k)^2$$
$$\mathbf{H}_k = \sqrt{\hat{\mathbf{H}}_k / (1-\beta^k)}$$

AdaGrad - Adaptive gradient method with $\mathbf{H}_k = \lambda_k \mathbf{I}$

- If $\mathbf{H}_k = \lambda_k \mathbf{I}$, it becomes stochastic gradient descent method with adaptive step-size $\frac{\alpha_k}{\lambda_k}$.

How step-size adapts?

If the stochastic gradient $\|\mathbf{g}^k\|$ is large/small $\rightarrow$ AdaGrad adjusts step-size α_k/λ_k smaller/larger

Adaptive gradient descent (AdaGrad with $\mathbf{H}_k = \lambda_k \mathbf{I}$) [25]

1. Set $Q^0 = 0$.
2. For $k = 0, 1, \ldots,$ iterate

$$\begin{cases} Q^k &= Q^{k-1} + \|\mathbf{g}^k\|^2 \\ \mathbf{H}_k &= \sqrt{Q^k} I \\ \mathbf{x}^{k+1} &= \mathbf{x}^k - \alpha_k \mathbf{H}_k^{-1} \mathbf{g}^k \end{cases}$$

AdaGrad - Adaptive gradient method with $\mathbf{H}_k = \lambda_k \mathbf{I}$

∘ If $\mathbf{H}_k = \lambda_k \mathbf{I}$, it becomes stochastic gradient descent method with adaptive step-size $\frac{\alpha_k}{\lambda_k}$.

How step-size adapts?

If the stochastic gradient $\|\mathbf{g}^k\|$ is large/small → AdaGrad adjusts step-size α_k/λ_k smaller/larger

Adaptive gradient descent (AdaGrad with $\mathbf{H}_k = \lambda_k \mathbf{I}$) [25]

1. Set Q^0 =0.
2. For $k = 0, 1, \ldots$, iterate

$$\begin{cases} Q^k & = Q^{k-1} + \|\mathbf{g}^k\|^2 \\ \mathbf{H}_k & = \sqrt{Q^k} I \\ \mathbf{x}^{k+1} & = \mathbf{x}^k - \alpha_k \mathbf{H}_k^{-1} \mathbf{g}^k \end{cases}$$

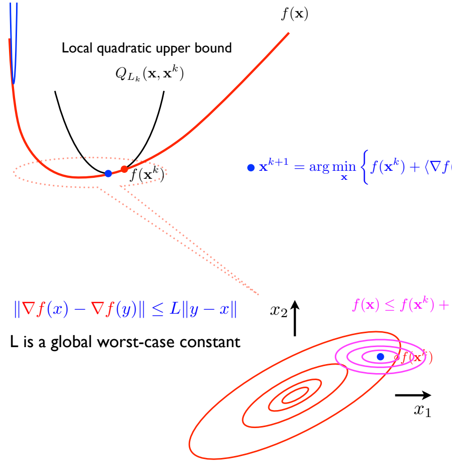
Adaptation through first-order information

- When $H_k = \lambda_k I$, AdaGrad estimates local geometry through stochastic gradient norms.
- Akin to estimating a local quadratic upper bound (majorization / minimization) using gradient history.

AdaGrad - Adaptive gradient method with $\mathbf{H}_k = \mathbf{D}_k$

Adaptation strategy with a positive diagonal matrix $\mathbf{D}_k$

Adaptive step-size + coordinate-wise extension = adaptive step-size for each coordinate



- $\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \left\{ f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle + \frac{L_k}{2}\|\mathbf{x} - \mathbf{x}^k\|_2^2 \right\}$

$$f(\mathbf{x}) \leq f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^T(\mathbf{x} - \mathbf{x}^k) + \frac{1}{2}\|\mathbf{x} - \mathbf{x}^k\|^2_{D_k^{-1}}$$

applies only locally

AdaGrad - Adaptive gradient method with $\mathbf{H}_k = \mathbf{D}_k$

- Suppose $\mathbf{H}_k$ is diagonal,

$$\mathbf{H}_k := \begin{bmatrix} \lambda_{k,1} & & 0 \\ & \ddots & \\ 0 & & \lambda_{k,d} \end{bmatrix},$$

- For each coordinate i, we have different step-size $\frac{\alpha_k}{\lambda_{k,i}}$ is the step-size.

Adaptive gradient descent(AdaGrad with $\mathbf{H}_k = \mathbf{D}_k$)

1. Set $\mathbf{Q}^0 = 0$.
2. For $k = 0, 1, \ldots,$ iterate

$$\begin{cases} \mathbf{Q}^k & = \mathbf{Q}^{k-1} + \mathrm{diag}(\mathbf{g}^k)^2 \\ \mathbf{H}_k & = \sqrt{\mathbf{Q}^k} \\ \mathbf{x}^{k+1} & = \mathbf{x}^k - \alpha_k \mathbf{H}_k^{-1} \mathbf{g}^k \end{cases}$$

AdaGrad - Adaptive gradient method with $\mathbf{H}_k = \mathbf{D}_k$

- Suppose $\mathbf{H}_k$ is diagonal,

$$\mathbf{H}_k := \begin{bmatrix} \lambda_{k,1} & & 0 \\ & \ddots & \\ 0 & & \lambda_{k,d} \end{bmatrix},$$

- For each coordinate i, we have different step-size $\frac{\alpha_k}{\lambda_{k,i}}$ is the step-size.

Adaptive gradient descent(AdaGrad with $\mathbf{H}_k = \mathbf{D}_k$)

1. Set $\mathbf{Q}^0 = 0$.
2. For $k = 0, 1, \ldots,$ iterate

$$\begin{cases} \mathbf{Q}^k &= \mathbf{Q}^{k-1} + \mathrm{diag}(\mathbf{g}^k)^2 \\ \mathbf{H}_k &= \sqrt{\mathbf{Q}^k} \\ \mathbf{x}^{k+1} &= \mathbf{x}^k - \alpha_k \mathbf{H}_k^{-1} \mathbf{g}^k \end{cases}$$

Adaptation across each coordinate

- When $\mathbf{H}_k = \mathbf{D}_k$, we adapt across each coordinate individually.
- Essentially, we have a finer treatment of the function we want to optimize.

RMSProp - Adaptive gradient method with $\mathbf{H}_k = \mathbf{D}_k$

What could be improved over AdaGrad?

1. Stochastic gradients have equal weights in step size.
2. Consider a *steep* function, flat around minimum $\rightarrow$ slow convergence at flat region.

RMSProp - Adaptive gradient method with $\mathbf{H}_k = \mathbf{D}_k$

What could be improved over AdaGrad?

1. Stochastic gradients have equal weights in step size.
2. Consider a *steep* function, flat around minimum $\rightarrow$ slow convergence at flat region.

AdaGrad with $\mathbf{H}_k = \mathbf{D}_k$
1. Set $\mathbf{Q}_0$ =0.   **2.** For $k = 0, 1, \ldots,$ iterate $\left\{ \begin{array}{ll} \mathbf{Q}^k & = \mathbf{Q}^{k-1} + \mathrm{diag}(\mathbf{g}^k)^2 \\ \mathbf{H}_k & = \sqrt{\mathbf{Q}^k} \\ \mathbf{x}^{k+1} & = \mathbf{x}^k - \alpha_k \mathbf{H}_k^{-1} \mathbf{g}^k \end{array} \right.$

RMSProp
1. Set $\mathbf{Q}_0$ =0.   **2.** For $k = 0, 1, \ldots,$ iterate $\left\{ \begin{array}{ll} \mathbf{Q}^k & = \beta \mathbf{Q}^{k-1} + (1-\beta)\mathrm{diag}(\mathbf{g}^k)^2 \\ \mathbf{H}_k & = \sqrt{\mathbf{Q}^k} \\ \mathbf{x}^{k+1} & = \mathbf{x}^k - \alpha_k \mathbf{H}_k^{-1} \mathbf{g}^k \end{array} \right.$

RMSProp - Adaptive gradient method with $\mathbf{H}_k = \mathbf{D}_k$

What could be improved over AdaGrad?

1. Stochastic gradients have equal weights in step size.
2. Consider a *steep* function, flat around minimum $\rightarrow$ slow convergence at flat region.

AdaGrad with $\mathbf{H}_k = \mathbf{D}_k$
1. Set $\mathbf{Q}_0 = 0$.
2. For $k = 0, 1, \ldots,$ iterate
$\begin{cases} \mathbf{Q}^k &= \mathbf{Q}^{k-1} + \mathrm{diag}(\mathbf{g}^k)^2 \\ \mathbf{H}_k &= \sqrt{\mathbf{Q}^k} \\ \mathbf{x}^{k+1} &= \mathbf{x}^k - \alpha_k \mathbf{H}_k^{-1}\mathbf{g}^k \end{cases}$

RMSProp
1. Set $\mathbf{Q}_0 = 0$.
2. For $k = 0, 1, \ldots,$ iterate
$\begin{cases} \mathbf{Q}^k &= \beta\mathbf{Q}^{k-1} + (1-\beta)\mathrm{diag}(\mathbf{g}^k)^2 \\ \mathbf{H}_k &= \sqrt{\mathbf{Q}^k} \\ \mathbf{x}^{k+1} &= \mathbf{x}^k - \alpha_k \mathbf{H}_k^{-1}\mathbf{g}^k \end{cases}$

- RMSProp uses weighted averaging with constant β
- Recent gradients have greater importance

AcceleGrad - Adaptive gradient + Accelerated gradient [26]

Motivation behind AcceleGrad

Is it possible to achieve acceleration when f is L-smooth, without knowing the Lipschitz constant?

AcceleGrad (Accelerated Adaptive Gradient Method)

Input : $\mathbf{x}^0 \in \mathcal{K}$, diameter D, weights $\{\alpha_k\}_{k\in\mathbb{N}}$, learning rate $\{\eta_k\}_{k\in\mathbb{N}}$

1. Set $\mathbf{y}^0 = \mathbf{z}^0 = \mathbf{x}^0$
2. For $k = 0, 1, \ldots,$ iterate

$$\begin{cases} \tau_k & := 1/\alpha_k \\ \mathbf{x}^{k+1} & = \tau_k \mathbf{z}^k + (1-\tau_k)\mathbf{y}^k, \text{define } \mathbf{g}_k := \nabla f(\mathbf{x}^{k+1}) \\ \mathbf{z}^{k+1} & = \Pi_{\mathcal{K}}(\mathbf{z}^k - \alpha_k \eta_k \mathbf{g}_k) \\ \mathbf{y}^{k+1} & = \mathbf{x}^{k+1} - \eta_k \mathbf{g}_k \end{cases}$$

Output : $\bar{\mathbf{y}}^k \propto \sum_{i=0}^{k-1} \alpha_i \mathbf{y}^{i+1}$

where $\Pi_{\mathcal{K}}(\mathbf{y}) = \arg\min_{\mathbf{x}\in\mathcal{K}} \langle \mathbf{x}-\mathbf{y}, \mathbf{x}-\mathbf{y}\rangle$ (projection onto $\mathcal{K}$).

***Remark:** ◦ This is essentially the **MD + GD** scheme [3], with an adaptive step size!

AcceleGrad - Properties and convergence

Learning rate and weight computation

Assume that function f has uniformly bounded gradient norms $\|\mathbf{g}^k\|^2 \leq G^2$, i.e., f is G-Lipschitz continuous. AcceleGrad uses the following weights and learning rate:

$$\alpha_k = \frac{k+1}{4}, \quad \eta_k = \frac{2D}{\sqrt{G^2 + \sum_{\tau=0}^{k} \alpha_\tau^2 \|\mathbf{g}^{\tau+1}\|^2}}$$

∘ Similar to RmsProp, AcceleGrad assignes greater weights to recent gradients.

Convergence rate of AcceleGrad

Assume that f is convex and L-smooth. Let $\mathcal{K}$ be a convex set with bounded diameter D, and assume $\mathbf{x}^\star \in \mathcal{K}$. Define $\bar{\mathbf{y}}^k = (\sum_{i=0}^{k-1} \alpha_i \mathbf{y}^{i+1})/(\sum_{i=0}^{k-1} \alpha_i)$. Then,

$$f(\overline{\mathbf{y}}^k) - f^\star \leq O\left(\frac{DG + LD^2 \log(LD/G)}{k^2}\right)$$

If f is only convex and G-Lipschitz, then

$$f(\overline{\mathbf{y}}^k) - f^\star \leq O\left(GD\sqrt{\log k}/\sqrt{k}\right)$$

ADAM - Adaptive moment estimation

Over-simplified idea of ADAM

RMSProp + 2nd order moment estimation = ADAM

ADAM - Adaptive moment estimation

Over-simplified idea of ADAM

RMSProp + 2nd order moment estimation = ADAM

ADAM
Input. Step size α, exponential decay rates $\beta_1, \beta_2 \in [0, 1)$
1. Set $\mathbf{m}_0, \mathbf{v}_0 = 0$ **2.** For $k = 0, 1, \ldots,$ iterate $\begin{cases} \mathbf{g}_k &= \nabla f(\mathbf{x}^{k-1}) \\ \mathbf{m}_k &= \beta_1 \mathbf{m}_{k-1} + (1-\beta_1)\mathbf{g}_k \leftarrow \text{1st order estimate} \\ \mathbf{v}_k &= \beta_2 \mathbf{v}_{k-1} + (1-\beta_2)\mathbf{g}_k^{.2} \leftarrow \text{2nd order estimate} \\ \hat{\mathbf{m}}_k &= \mathbf{m}_k/(1-\beta_1^k) \leftarrow \text{Bias correction} \\ \hat{\mathbf{v}}_k &= \mathbf{v}_k/(1-\beta_2^k) \leftarrow \text{Bias correction} \\ \mathbf{H}_k &= \sqrt{\hat{\mathbf{v}}_k} + \epsilon \\ \mathbf{x}^{k+1} &= \mathbf{x}^k - \alpha \hat{\mathbf{m}}_k./\mathbf{H}_k \end{cases}$
Output : $\mathbf{x}^k$

(Every vector operation is an element-wise operation)

Non-convergence of ADAM and a new method: AmsGrad

- It has been shown that ADAM may not converge for *some* objective functions [41].
- An ADAM alternative is proposed that is proved to be convergent [32].

AmsGrad

Input. Step size $\{\gamma_k\}_{k\in\mathbb{N}}$, exponential decay rates $\{\beta_{1,k}\}_{k\in\mathbb{N}}$, $\beta_2 \in [0,1)$

1. Set $\mathbf{m}_0 = 0, \mathbf{v}_0 = 0$ and $\hat{\mathbf{v}}_0 > 0$

2. For $k = 1, 2, \ldots,$ iterate

$$\begin{cases} \mathbf{g}_k & = G(\mathbf{x}^k, \theta) \\ \mathbf{m}_k & = \beta_{1,k}\mathbf{m}_{k-1} + (1-\beta_{1,k})\mathbf{g}_k \leftarrow \text{1st order estimate} \\ \mathbf{v}_k & = \beta_2 \mathbf{v}_{k-1} + (1-\beta_2)\mathbf{g}_k^{.2} \leftarrow \text{2nd order estimate} \\ \hat{\mathbf{v}}_k & = \max\{\hat{\mathbf{v}}_{k-1}, \mathbf{v}_k\} \text{ and } \hat{\mathbf{V}}_k = \text{diag}(\hat{\mathbf{v}}_k) \\ \mathbf{H}_k & = \sqrt{\hat{\mathbf{v}}_k} \\ \mathbf{x}^{k+1} & = \Pi_{\mathcal{X}}^{\sqrt{\hat{\mathbf{V}}_k}}(\mathbf{x}^k - \gamma_k \hat{\mathbf{m}}_k./\mathbf{H}_k) \end{cases}$$

Output : $\mathbf{x}^k$

where $\Pi_{\mathcal{K}}^{\mathbf{A}}(\mathbf{y}) = \arg\min_{\mathbf{x}\in\mathcal{K}} \langle(\mathbf{x}-\mathbf{y}), \mathbf{A}(\mathbf{x}-\mathbf{y})\rangle$ (weighted projection onto $\mathcal{K}$).
(Every vector operation is an element-wise operation)

AdaGrad & AmsGrad for non-convex optimization

Theorem (AdaGrad convergence rate: stochastic, non-convex [37])

Assume f is non-convex and L-smooth, such that $\|\nabla f(\mathbf{x})\|^2 \leq G^2$ and $f^\star = \inf_{\mathbf{x}} f(\mathbf{x}) > \infty$. Also consider bounded variance for unbiased gradient estimates, i.e., $\mathbb{E}\left[\|G(\mathbf{x},\theta) - \nabla f(\mathbf{x})\|^2 | \mathbf{x}\right] \leq \sigma^2$. Then with probability $1-\delta$,

$$\min_{i \in \{1,..,k-1\}} \|\nabla f(\mathbf{x}^i)\|^2 = \tilde{\mathcal{O}}\left(\frac{\sigma}{\delta^{3/2}\sqrt{k}}\right)$$

◦ **Note:** As $1-\delta \to 1$, the rate deteriorates by a factor of $\delta^{-3/2}$.

Theorem (AmsGrad convergence rate 1: stochastic, non-convex [7])

Let $\mathbf{g}_k = G(\mathbf{x}^k, \theta)$. Assume $\|\mathbf{g}_k\| \leq G$. Consider a non-increasing sequence $\beta_{1,k}$ and $\beta_{1,k} \leq \beta_1 \in [0,1)$. Set $\gamma_k = 1/\sqrt{k}$. Then,

$$\min_{i \in \{1,..,k-1\}} \mathbb{E}\left[\|\nabla f(\mathbf{x}^i)\|^2\right] = O\left(\frac{\log k}{\sqrt{k}}\right).$$

AdaGrad & AmsGrad for non-convex optimization

Theorem (AdaGrad convergence rate: stochastic, non-convex [37])

Assume f is non-convex and L-smooth, such that $\|\nabla f(\mathbf{x})\|^2 \leq G^2$ and $f^\star = \inf_{\mathbf{x}} f(\mathbf{x}) > \infty$. Also consider bounded variance for unbiased gradient estimates, i.e., $\mathbb{E}\left[\|G(\mathbf{x},\theta) - \nabla f(\mathbf{x})\|^2 | \mathbf{x}\right] \leq \sigma^2$. Then with probability $1-\delta$,

$$\min_{i\in\{1,..,k-1\}} \|\nabla f(\mathbf{x}^i)\|^2 = \tilde{\mathcal{O}}\left(\frac{\sigma}{\delta^{3/2}\sqrt{k}}\right)$$

- **Note:** As $1-\delta \to 1$, the rate deteriorates by a factor of $\delta^{-3/2}$.

Theorem (AmsGrad convergence rate 2: stochastic, non-convex [43, 6])

Consider $f : \mathbb{R}^p \to \mathbb{R}$ to be non-convex and L-smooth. Assume $\|G(\mathbf{x},\theta)\|_\infty \leq G_\infty$ and set $\gamma_k = 1/\sqrt{pT}$. Also define $\mathbf{x}_{out} = \mathbf{x}^k$, for $k = 1, \ldots, T$ with probability $\gamma_k / \sum_{i=1}^T \gamma_i$. Then,

$$\mathbb{E}\left[\|\nabla f(\mathbf{x}_{out})\|^2\right] = \mathcal{O}\left(\sqrt{\frac{p}{T}}\right).$$

Adam variants without large batch sizes

Guarantees of Adam-variants [1]

By using one subgradient each iteration, with the same setup as before, AMSGrad converges for $\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$

$$\mathbb{E}\|G_\lambda(\mathbf{x}_{\text{out}})\|^2 \leq \tilde{\mathcal{O}}\left(\sqrt{\frac{1}{T}}\right), \tag{2}$$

on the gradient mapping $G_\lambda(\mathbf{x}) = \frac{\mathbf{H}_k^{1/2}}{\lambda}\left(\mathbf{x} - P_{\mathcal{X}}^{\mathbf{H}_k}(\mathbf{x} - \lambda\mathbf{H}_k^{-1}\nabla f(\mathbf{x}))\right)$, where $\mathbf{x}_{\text{out}}$ is chosen uniformly at random from the iterates.

ADAM vs. AcceleGrad

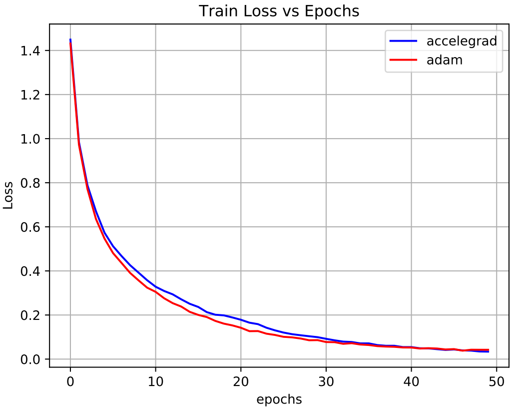


Figure: Resnet classifier optimization (train loss)

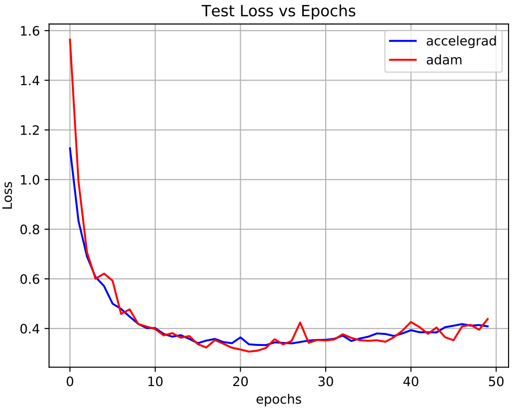


Figure: Resnet classifier optimization (test loss)

Performance of optimization algorithms (nonconvex)

- Assuming only L-smoothness, SGD, Adagrad, RmsProp, ADAM & AmsGrad and Accelegrad has $\frac{1}{\sqrt{k}}$-rate
- Additional assumptions help improve this rate
  - Polyak-Lojasiewicz (PL)[1]
  - Strong growth condition (SGC)[2]

[1] J. Bolte, T. P. Nguyen, J. Peypouquet, and B. W. Suter. "From error bounds to the complexity of first-order descent methods for convex functions."

[2] V. Cevher and B. C. Vu. "On the linear convergence of the stochastic gradient method with constant step-size."

Performance of optimization algorithms (nonconvex)

◦ Assuming only L-smoothness, SGD, Adagrad, RmsProp, ADAM & AmsGrad and Accelegrad has $\frac{1}{\sqrt{k}}$-rate

◦ Additional assumptions help improve this rate

- Polyak-Lojasiewicz (PL)[1]
- Strong growth condition (SGC)[2]

◦ A non-exhaustive comparison:

Assumptions on f	**Algorithm**	**Convergence rate**	**Iteration complexity**
L-smooth	Basically all first order methods	Sublinear $(1/\sqrt{k})$	One stochastic gradient
L-smooth + SGC	SGD	Sublinear $(1/k)$[36]	One stochastic gradient
L-smooth + SGC + PL	SGD	Linear (ρk) [36]	One stochastic gradient

[1] J. Bolte, T. P. Nguyen, J. Peypouquet, and B. W. Suter. "From error bounds to the complexity of first-order descent methods for convex functions."

[2] V. Cevher and B. C. Vu. "On the linear convergence of the stochastic gradient method with constant step-size."

Implicit regularization of adaptive methods may overfit

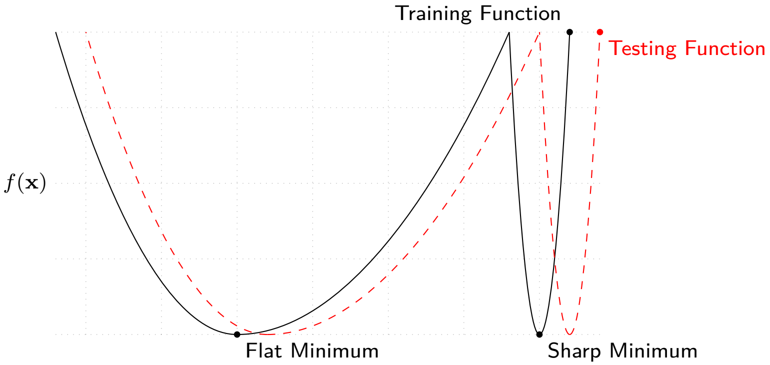


Figure: Sharp Minima vs Flat Minima [22]

- Intuition suggests flat minima has better generalization property than sharp minima
- Empirically, adaptive methods finds sharper minima than ones found by SGD
- The relationship between sharpness of minima and their generalization is open [8, 12]

Example: Generalization performance

◦ Adaptive learning methods may converge fast but generalize worse

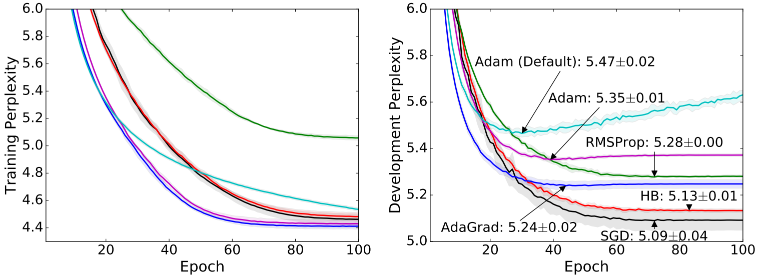


Figure: Performance of different optimizers in training and development set of a language modeling problem. The training and test perplexity are the exponential values of training and test losses.[39]

Neural Network Architectures

- Deeper and more complicated models correlates with better performance
- No universal optimizers other than slow and steady SGD
- A long way to go (makes it exciting)...

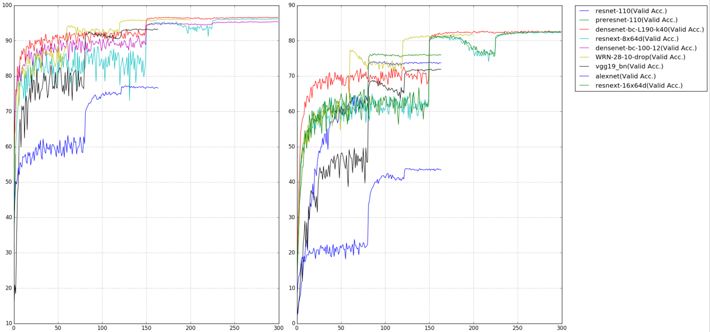


Figure: Performance of popular architectures on test set in CIFAR10 (left) and CIFAR100 (right). [3]

[3] Credit to: https://github.com/bearpaw/pytorch-classification

Wrap up!

- Recitation on Friday!

*Perturbed SGD escapes saddle points

Theorem (Convergence of PSGD [13])

Suppose that f has the following properties

- f *is an* $(\alpha, \gamma, \epsilon, \delta)$*-strict saddle,*
- f *is* β*-smooth.*
- *its Hessian is* ρ*-Lipschitz. i.e.* $\left\| \nabla^2 f(\mathbf{x}) - \nabla^2 f(\mathbf{y}) \right\| \leq \rho \left\| \mathbf{x} - \mathbf{y} \right\|$.

Then there exists a threshold $\gamma_{\max}$ *such that by choosing*

- $\gamma \leq \gamma_{\max} / \max\{1, \log(1/\zeta)\}$
- $T = O(\gamma^{-2} \log(1/\zeta))$.

*the algorithm **Perturbed SGD** outputs with probability at least* $1 - \zeta$ *a point* $\mathbf{x}_T$ *that is* $O(\sqrt{\gamma \log(1/\gamma\zeta)})$ *close to some local minimum* $\mathbf{x}^\star$.

*Convergence of SGD in non-convex problems with small step-size

Assumptions

1. Function f is lower bounded: $\exists f^\star$ s.t. $\forall \mathbf{x} \in \mathcal{X}, f(\mathbf{x}) \geq f^\star$
2. Function f has Lipschitz continuous gradient:

$$\|\nabla f(\mathbf{x}_1) - \nabla f(\mathbf{x}_2)\|_2 \leq L\|\mathbf{x}_1 - \mathbf{x}_2\|_2 \tag{3}$$

3. The stochastic gradient $\hat{\mathbf{g}}_\mathbf{x}$ is unbiased and has bounded variance:

$$\mathbb{E}(\hat{\mathbf{g}}) = \mathbf{g}, \quad \mathbb{E}(\|\hat{\mathbf{g}} - \mathbf{g}\|_2^2) \leq \sigma^2 \tag{4}$$

Theorem (Convergence of SGD in non-convex problems [5])

For SGD with assumptions above, N iterations and stepsize $\gamma_t = \frac{1}{L\sqrt{N}}$, we have

$$\mathbb{E}\left[\frac{1}{N}\sum_{t=0}^{N-1}\|\mathbf{g}^t\|_2^2\right] \sim \mathcal{O}\left(\frac{1}{\sqrt{N}}\right), \tag{5}$$

where the convergence is captured by the gradient norm.

⋆Convergence of SGD

Proof

Take the assumption 2 and algorithmic update policy $\mathbf{x}^{t+1} = \mathbf{x}^t - \gamma\hat{\mathbf{g}}^t$

$$\begin{aligned} f(\mathbf{x}_{t+1}) - f(\mathbf{x}_t) &\leq (\mathbf{x}_{t+1} - \mathbf{x}_t)^T \mathbf{g}^t + \frac{L}{2}\|\mathbf{x}_{t+1} - \mathbf{x}_t\|_2^2 \\ &= -\gamma_t (\hat{\mathbf{g}}^t)^T \mathbf{g}^t + \frac{\gamma_t^2 L}{2}\|\hat{\mathbf{g}}^t\|_2^2 \end{aligned} \tag{6}$$

Take the expectation and use the assumption 3

$$\mathbb{E}[f(\mathbf{x}_{t+1}) - f(\mathbf{x}_t)] = -\gamma_t \|\mathbf{g}^t\|_2^2 + \frac{\gamma_t^2 L}{2}(\|\mathbf{g}^t\|_2^2 + \sigma^2) \tag{7}$$

Set the learning rate $\gamma_t = \frac{1}{L\sqrt{N}}$

$$\begin{aligned} \mathbb{E}[f(\mathbf{x}_{t+1}) - f(\mathbf{x}_t)] &= -\frac{1}{L\sqrt{N}}\|\mathbf{g}^t\|_2^2 + \frac{1}{2LN}(\|\mathbf{g}^t\|_2^2 + \sigma^2) \\ &\leq -\frac{1}{2L\sqrt{N}}\|\mathbf{g}^t\|_2^2 + \frac{\sigma^2}{2LN} \end{aligned} \tag{8}$$

*Convergence of SGD

Proof (Cont'd).

Sum the inequality of N steps together and use assumption 1

$$\begin{aligned} f(\mathbf{x}_0) - f^\star &\geq f(\mathbf{x}_0) - \mathbb{E}[f(\mathbf{x}_N)] \\ &= \mathbb{E}\left[\sum_{t=0}^{N-1} (f(\mathbf{x}_t) - f(\mathbf{x}_{t+1}))\right] \\ &\geq \frac{1}{2L}\mathbb{E}\left[\sum_{t=0}^{N-1} (\frac{\|\mathbf{g}^t\|_2^2}{\sqrt{N}} - \frac{\sigma^2}{N})\right] \end{aligned} \tag{9}$$

Rearrange the inequality, we have the following

$$\mathbb{E}\left[\frac{1}{N}\sum_{t=0}^{N-1} \|\mathbf{g}^t\|_2^2\right] \leq \frac{1}{\sqrt{N}}[2L(f(\mathbf{x}_0) - f^\star + \sigma^2)] \tag{10}$$

The right hand side vanishes as $N \to \infty$, so $\mathbb{E}\left[\frac{1}{N}\sum_{t=0}^{N-1} \|\mathbf{g}^t\|_2^2\right]$ vanishes also. This indicates the model converges to a critical point. □

References I

[1] Ahmet Alacaoglu, Yura Malitsky, and Volkan Cevher.
Convergence of adaptive algorithms for weakly convex constrained optimization, 2020.

[2] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song.
A convergence theory for deep learning via over-parameterization.
In *International Conference on Machine Learning*, pages 242–252. PMLR, 2019.

[3] Zeyuan Allen-Zhu and Lorenzo Orecchia.
Linear Coupling: An Ultimate Unification of Gradient and Mirror Descent.
In *Proceedings of the 8th Innovations in Theoretical Computer Science*, ITCS '17, 2017.
Full version available at `http://arxiv.org/abs/1407.1537`.

[4] Michel Benaïm.
Dynamics of stochastic approximation algorithms.
In Jacques Azéma, Michel Émery, Michel Ledoux, and Marc Yor, editors, *Séminaire de Probabilités XXXIII*, volume 1709 of *Lecture Notes in Mathematics*, pages 1–68. Springer Berlin Heidelberg, 1999.

[5] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Anima Anandkumar.
signsgd: compressed optimisation for non-convex problems.
arXiv preprint arXiv:1802.04434, 2018.

References II

[6] Jinghui Chen, Dongruo Zhou, Yiqi Tang, Ziyan Yang, Yuan Cao, and Quanquan Gu.
Closing the generalization gap of adaptive gradient methods in training deep neural networks.
In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3267–3275. International Joint Conferences on Artificial Intelligence Organization, 7 2020.
Main track.

[7] Xiangyi Chen, Sijia Liu, Ruoyu Sun, and Mingyi Hong.
On the convergence of a class of adam-type algorithms for non-convex optimization.
In *International Conference on Learning Representations*, 2019.

[8] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio.
Sharp minima can generalize for deep nets.
arXiv preprint arXiv:1703.04933, 2017.

[9] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai.
Gradient descent finds global minima of deep neural networks.
In *International Conference on Machine Learning*, pages 1675–1685, 2019.

[10] Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh.
Gradient descent provably optimizes over-parameterized neural networks.
arXiv preprint arXiv:1810.02054, 2018.

References III

[11] John Duchi, Elad Hazan, and Yoram Singer.
Adaptive subgradient methods for online learning and stochastic optimization.
Journal of Machine Learning Research, 12(Jul):2121–2159, 2011.

[12] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur.
Sharpness-aware minimization for efficiently improving generalization, 2020.

[13] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan.
Escaping from saddle points—online stochastic gradient for tensor decomposition.
In *Conference on Learning Theory*, pages 797–842, 2015.

[14] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan.
Escaping from saddle points — Online stochastic gradient for tensor decomposition.
In *COLT '15: Proceedings of the 28th Annual Conference on Learning Theory*, 2015.

[15] Saeed Ghadimi and Guanghui Lan.
Stochastic first- and zeroth-order methods for nonconvex stochastic programming.
SIAM Journal on Optimization, 23(4):2341–2368, 2013.

References IV

[16] Guang-Bin Huang and H. A. Babri.
Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions.
IEEE Transactions on Neural Networks, 9(1):224–229, 1998.

[17] Benjamin D Haeffele and René Vidal.
Global optimality in neural network training.
In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7331–7339, 2017.

[18] Moritz Hardt and Tengyu Ma.
Identity matters in deep learning.
arXiv preprint arXiv:1611.04231, 2016.

[19] Guang-Bin Huang.
Learning capability and storage capacity of two-hidden-layer feedforward networks.
IEEE Transactions on Neural Networks, 14(2):274–281, 2003.

[20] S. . Huang and Y. . Huang.
Bounds on the number of hidden neurons in multilayer perceptrons.
IEEE Transactions on Neural Networks, 2(1):47–55, 1991.

References V

[21] Kenji Kawaguchi and Jiaoyang Huang.
Gradient descent finds global minima for generalizable deep neural networks of practical sizes.
In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 92–99. IEEE, 2019.

[22] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang.
On large-batch training for deep learning: Generalization gap and sharp minima.
arXiv preprint arXiv:1609.04836, 2016.

[23] Diederik Kingma and Jimmy Ba.
Adam: A method for stochastic optimization.
arXiv preprint arXiv:1412.6980, 2014.

[24] Jason D. Lee, Ioannis Panageas, Georgios Piliouras, Max Simchowitz, Michael I. Jordan, and Benjamin Recht.
First-order methods almost always avoid strict saddle points.
Mathematical Programming, 176(1):311–337, February 2019.

[25] Kfir Levy.
Online to offline conversions, universality and adaptive minibatch sizes.
In *Advances in Neural Information Processing Systems*, pages 1613–1622, 2017.

References VI

[26] Kfir Levy, Alp Yurtsever, and Volkan Cevher.
Online adaptive methods, universality and acceleration.
In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018.

[27] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein.
Visualizing the Loss Landscape of Neural Nets.
arXiv, Dec 2017.

[28] Yuanzhi Li and Yingyu Liang.
Learning overparameterized neural networks via stochastic gradient descent on structured data.
In *Advances in Neural Information Processing Systems*, pages 8157–8166, 2018.

[29] Lennart Ljung.
Analysis of recursive stochastic algorithms.
22(4):551–575, August 1977.

[30] Panayotis Mertikopoulos, Nadav Hallak, Ali Kavis, and Volkan Cevher.
On the almost sure convergence of stochastic gradient descent in non-convex problems, 2020.

[31] Quynh Nguyen and Matthias Hein.
Optimization landscape and expressivity of deep cnns.
In *International conference on machine learning*, pages 3730–3739. PMLR, 2018.

References VII

[32] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar.
On the convergence of adam and beyond.
arXiv preprint arXiv:1904.09237, 2019.

[33] Gregory Roth and W. Sandholm.
Stochastic approximations with constant step size and differential inclusions.
SIAM J. Control. Optim., 51:525–555, 2013.

[34] Zhao Song and Xin Yang.
Quadratic suffices for over-parametrization via matrix chernoff bound.
arXiv preprint arXiv:1906.03593, 2019.

[35] Tijmen Tieleman and Geoffrey Hinton.
Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude.
COURSERA: Neural networks for machine learning, 4(2):26–31, 2012.

[36] Sharan Vaswani, Francis Bach, and Mark Schmidt.
Fast and faster convergence of sgd for over-parameterized models and an accelerated perceptron, 2019.

References VIII

[37] Rachel Ward, Xiaoxia Wu, and Leon Bottou.
AdaGrad stepsizes: Sharp convergence over nonconvex landscapes.
In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6677–6686, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[38] Max Welling and Yee W Teh.
Bayesian learning via stochastic gradient langevin dynamics.
In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688, 2011.

[39] Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht.
The marginal value of adaptive gradient methods in machine learning.
In *Advances in Neural Information Processing Systems*, pages 4148–4158, 2017.

[40] Chulhee Yun, Suvrit Sra, and Ali Jadbabaie.
Small relu networks are powerful memorizers: a tight analysis of memorization capacity.
In *Advances in Neural Information Processing Systems*, pages 15558–15569, 2019.

[41] Manzil Zaheer, Sashank Reddi, Devendra Sachan, Satyen Kale, and Sanjiv Kumar.
Adaptive methods for nonconvex optimization.
In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 9793–9803. Curran Associates, Inc., 2018.

References IX

[42] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals.
Understanding deep learning requires rethinking generalization.
arXiv preprint arXiv:1611.03530, 2016.

[43] Dongruo Zhou, Yiqi Tang, Ziyan Yang, Yuan Cao, and Quanquan Gu.
On the convergence of adaptive gradient methods for nonconvex optimization.
ArXiv, abs/1808.05671, 2018.

[44] Difan Zou, Yuan Cao, Dongruo Zhou, and Quanquan Gu.
Gradient descent optimizes over-parameterized deep relu networks.
Machine Learning, 109(3):467–492, 2020.

[45] Difan Zou and Quanquan Gu.
An improved analysis of training over-parameterized deep neural networks.
In *Advances in Neural Information Processing Systems*, pages 2055–2064, 2019.